APPENDIX I

```
1:      <!--- Copyright 1999 WebTrends Corporation --->
2:      <!--- http://www.webtrends.com --->
3:      <!--- Modification of this code is not allowed and will permanently disable your
account --->
4:      <script language="JavaScript1.2">
5:      <!---
6:      var code = "";
7:      var ORDER = "<% ORDER %>"
        var SERVER = "";
8:      var title = escape(document.title);
9:      var url = window.document.URL;
10:     var orderstr = escape(order);
11:     var get = "http://stats.webtrendslive.com/scripts/enterprise.cgi";
12:     get += "?sid=000-99-9-7-27-7349&siteID=232";
13:     get += "&title=" + title + "&url=" + url;
16:     document.write("<" + "script src='" + get + "'></script>");
17:     //-->
18:     </script>
19:     <script language="JavaScript1.2">
20:     document.write(code);
21:     document.write("<" + "!---"); </script>
22:     <img src="http://stats.webtrendslive.com/scripts/enterprise3.cgi?sid=000-99-
9-7-27-
23:     7349&siteID=232&url=">
24:     <script language="JavaScript1.2">
25:     document.write(" ---" + ">");
26:     </script>
27:     <noscript>
28:     <img src="http://stats.webtrendslive.com/scripts/enterprise3.cgi?sid=000-99-
9-7-27-29:      7349&siteID=232&url=">
30:     </noscript>
31:     <!--- End of WebTrends Counter insertion --->
```

APPENDIX II

```
<%
ORDER = "D1;"
FOR i = 0 to UBOUND(orders)
ORDER = ORDER + product(i) & "," & category(i) >>
& "," & number_sold(i) & "," & unit_price(i) >>
& ";"
NEXT
%>
```

('>>' indicates line continues)

APPENDIX III

```
<!-- START OF WEBTRENDS LIVE TAG INSERTION -->
<!-- Copyright 1999-2000 WebTrends Corporation -->

<!-- Visit our corporate website at http://www.webtrends.com -->
<!-- Visit our Webtrends Live website at http://www.webtrendslive.com -->
<!-- eCommerce Revenue Tracking (patent pending) -->
<SCRIPT LANGUAGE="JavaScript1.1">
var ORDER= "";
var SERVER= "";
var CONTENTGROUP= "";

// You may customize the values of the above three variables to suit your website.
// Simply insert your own values between the double-quotes.
// The lines below show some examples:
//   var ORDER= "D2,Business to Consumer,Pocket FM Radio,Audio Products,10,499.99;";
//   var SERVER= "Name of this web server";
//   var CONTENTGROUP = "Content group name for this page";
</SCRIPT>

<!-- Modification of this code is not allowed and will permanently disable your account! -->

<SCRIPT LANGUAGE="JavaScript1.1">
v = '<' + 'SCRIPT SRC="';
v+= 'http://stats.webtrendslive.com/S005-00-5-18-2994-11462/scripts/wttagv2.cgi?sid=005-
00-5-18-2994-11462&siteID=11462&tagver=2&tz=-800&ed=ecommerce&button=&';
v+= 'order=' + escape(ORDER) + '&';
v+= 'server=' + escape(SERVER) + '&';
v+= 'url=' + escape(window.document.URL) + '&';
v+= 'ref=' + escape(window.document.referrer) + '&';
v+= 'title=' + escape(document.title) + '&';
v+= '"' + '>' + '<' + '/' + 'SCRIPT' + '>';
document.write( v);
</SCRIPT>

<NOSCRIPT>
<IMG SRC="http://stats.webtrendslive.com/S005-00-5-18-2994-
11462/scripts/wttagv2_ns.cgi?uid=005-00-5-18-2994-11462&siteID=11462&tagver=2&tz=-
800&ed=ecommerce&button=&javaOk=No">
</NOSCRIPT>

<!-- END OF WEBTRENDS LIVE TAG INSERTION -->
```

APPENDIX IV

==========

==

This class object stores (as part of it) the data for the configured filters.

```
class CLogStats
{
public:
        CLogStats();
        ~CLogStats();

public:
        enum ENUM_STATE
        {
                ENUM_UNUSED= 0,

                ENUM_ANALYZE= 1,
                ENUM_ANALYZE_BUSY= 2,
                ENUM_DATABASE= 3,
                ENUM_DATABASE_BUSY= 4,

        };

public:
        ENUM_STATE flnuse;

        DWORD dwStartTick;
        DWORD dwCountTick;

        char szLogfile[MAX_PATH];
        unsigned int nLogfile_size;
        unsigned int nLogfile_read;
        FILETIME ftCreateOn;

        char szComment[MAX_PATH];
        DWORD dwSiteCountTick;

        int nHits;
        int nDNSBlanks;
        int nDNSResolved;
        int nDNSUnresolved;

        FILETIME ftFirstHit;

public:
        char szDsn[MAX_PATH];

public:
```

```
        int nUid;
        int nSiteId;

        int nTimeZone;
        char szCategory[100];
        char szSubCategory[100];
        char szLastAccess[20];
        char szHomePage[100];

public:
        char szIncludeFilter[400];
        char szExcludeFilter[400];

        struct IPFILTER
        {
        public:
                unsigned int start[4];
                unsigned int end[4];
        public:
                IPFILTER()
                {
                        memset( start, 0, sizeof(start));
                        memset( end, 0, sizeof(end));
                }
        };

        std::list< IPFILTER> listExcludeFilter;
        std::list< IPFILTER> listIncludeFilter;

public:
        char szDBLocation[100];
        char szSiteVer[100];

public:
        Table* pdb;
        Table* pGlobalDB;

public:
        SiteTables* pSiteTables;

        GlobalTables* pGlobalTables;

public:
        DWORD dwGlobalStartTick;

public:
        CLogStats* pNode_Global;
};
```

==========

==

This function reads from the database the configured filters and parse it out for use later.

```
int CheckSiteId( Table* pGlobalDB, CLogStats* pSite)
{
	if( pGlobalDB->hdbc)
	{
		TableStmt hstmt( pGlobalDB->hdbc);
		int rc;

		//rc= db.ExecDirect( hstmt, "SELECT timezone, sitecategory,
sitesubcategory, homepage, firstaccess FROM SiteSettings WHERE siteid=%u", pSite-
>nSiteId);
		rc= pGlobalDB->ExecDirect( hstmt,
			"SELECT DBLocation, SiteVer, timezone, sitecategory,
sitesubcategory, homepage, firstaccess, includefilter, excludefilter "
			"FROM PageCounterSites "
			"WHERE siteid=%u",
			pSite->nSiteId);
		if( rc== SQL_SUCCESS)
		{
			char szFirstAccess[100]= "";

			pSite->nTimeZone= 0;
			pSite->szCategory[0]= 0;
			pSite->szSubCategory[0]= 0;
			pSite->szHomePage[0]= 0;
			pSite->szIncludeFilter[0]= 0;
			pSite->szExcludeFilter[0]= 0;

			SDWORD nLen[30];
			int nCol= 0;

			nLen[nCol]= 0;
			rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, &pSite-
>szDBLocation, sizeof(pSite->szDBLocation), &nLen[nCol]);
			nCol++;

			nLen[nCol]= 0;
			rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, &pSite-
>szSiteVer, sizeof(pSite->szSiteVer), &nLen[nCol]);
			nCol++;

			nLen[nCol]= 0;
			rc= SQLBindCol( hstmt, nCol+1, SQL_C_LONG, &pSite-
>nTimeZone, sizeof(pSite->nTimeZone), &nLen[nCol]);
```

```
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, (void*) &pSite-
>szCategory, sizeof(pSite->szCategory), &nLen[nCol]);
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, (void*) &pSite-
>szSubCategory, sizeof(pSite->szSubCategory), &nLen[nCol]);
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, (void*) &pSite-
>szHomePage, sizeof(pSite->szHomePage), &nLen[nCol]);
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, szFirstAccess,
sizeof(szFirstAccess), &nLen[nCol]);
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, pSite-
>szIncludeFilter, sizeof(pSite->szIncludeFilter), &nLen[nCol]);
                        nCol++;

                        nLen[nCol]= SQL_NTS;
                        rc= SQLBindCol( hstmt, nCol+1, SQL_C_CHAR, pSite-
>szExcludeFilter, sizeof(pSite->szExcludeFilter), &nLen[nCol]);
                        nCol++;

                        rc= SQLFetch( hstmt);
                        if( rc== SQL_SUCCESS || rc== SQL_SUCCESS_WITH_INFO)
                        {
                                //=================================

                                // Parse out the Include Filter

                                if( pSite->szIncludeFilter[0])
                                {
                                        pSite->listIncludeFilter.clear();

                                        CLogStats::IPFILTER node;

                                        int n= 0;
                                        char* p= pSite->szIncludeFilter;
                                        while( p && p[0])
                                        {
                                                if( p[0]== '*')
```

```
                                        {
                                                node.start[n]= 0;
                                                node.end[n]= 255;
                                                n++;

                                                p++;
                                        }
                                        else
                                        {
                                                node.start[n]= atoi( p);
                                                p+= strspn( p, "0123456789");

                                                if( p[0]== '-')
                                                {
                                                        p++;

                                                        node.end[n]= atoi( p);
                                                        p+= strspn( p, "0123456789");
                                                }
                                                else
                                                {
                                                        node.end[n]= node.start[n];
                                                }

                                                n++;
                                        }
                                        if( p[0]== '.')
                                        {
                                                p++;
                                        }
                                        else
                                        {
                                                if( n== 4)
                                                {
                                                        pSite-
>listIncludeFilter.push_back( node);
                                                }
                                        }

                                        if( n>= 4)
                                        {
                                                p+= strcspn( p, "; ");
                                                p+= strspn( p, "; ");

                                                n= 0;
                                        }
                                }
                        }

                        //================================
```

```
// Parse out the Exclude Filter

if( pSite->szExcludeFilter[0])
{
	pSite->listExcludeFilter.clear();

	CLogStats::IPFILTER node;

	int n= 0;
	char* p= pSite->szExcludeFilter;
	while( p && p[0])
	{
		if( p[0]== '*')
		{
			node.start[n]= 0;
			node.end[n]= 255;
			n++;

			p++;
		}
		else
		{
			node.start[n]= atoi( p);
			p+= strspn( p, "0123456789");

			if( p[0]== '-')
			{
				p++;

				node.end[n]= atoi( p);
				p+= strspn( p, "0123456789");
			}
			else
			{
				node.end[n]= node.start[n];
			}

			n++;
		}
		if( p[0]== '.')
		{
			p++;
		}
		else
		{
			if( n== 4)
			{
				pSite-
>listExcludeFilter.push_back( node);
```

```
                                        }
                                }

                                if( n>= 4)
                                {
                                        p+= strcspn( p, "; ");
                                        p+= strspn( p, "; ");

                                        n= 0;
                                }
                        }
                }

                //=================================

                if( !szFirstAccess[0])
                {
                        TableStmt hstmt( pGlobalDB->hdbc);
                        rc= pGlobalDB->ExecDirect( hstmt,
                                "UPDATE [PageCounterSites] SET
[firstaccess]=GETDATE() WHERE siteid=%u",
                                pSite->nSiteId,
                                0);
                }

                //=================================

                strlwr( pSite->szDBLocation);

                if( 1)
                {
                        std::map< std::string, std::string>::iterator i;
                        i= gGlobalSettings.zSiteDsn.find( pSite-
>szDBLocation);
                        if( i!= gGlobalSettings.zSiteDsn.end())
                        {
                                strcpy( pSite->szDsn, (*i).second.c_str());

                                return 0;
                        }
                }

                //=================================

                dprintf( "DSN '%s' not found\r\n", pSite->szDBLocation);
        }
        else
        {
                dprintf( "CheckSiteId fetch failed with rc=%u\r\n", rc);
        }
```

```
		}
		else
		{
			dprintf( "CheckSiteId failed with rc=%u, msg=%s\r\n", rc,
pGlobalDB->szErrorMsg);
		}
	}

	pSite->nSiteId= -1;
	return -1;
}
```

==========

==

This function is run on every hit and the filter logic is executed to determine
if this hit should be counted.

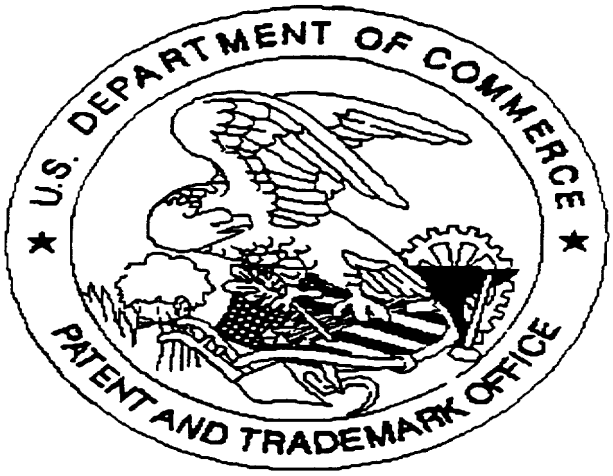
```
int SiteCount( Table* pdb, Table* pGlobalDB, CLogStats* pStats, CLogEntry* pLogEntry)
{
	DWORD dwStartTick= GetTickCount();

	// Make sure we look at tags for the right version!
	if( atoi( pLogEntry->pTagVersion)!= 1)
	{
		DWORD dwElapsed= GetTickCount() - dwStartTick;
		pStats->dwSiteCountTick+= dwElapsed;

		return 0;
	}

	int id;

	//=====================================

	// Check Include Filters
	if( 1)
	{
		if( pStats->listIncludeFilter.size())
		{
			unsigned long l;
			unsigned char* p= (unsigned char*) &l;

			l= inet_addr( pLogEntry->pHost);

			std::list< CLogStats::IPFILTER>::iterator i;
			for( i= pStats->listIncludeFilter.begin(); i!= pStats-
>listIncludeFilter.end(); i++)
```

```
                        {
                                CLogStats::IPFILTER& n= (*i);

                                if( p[0]>= n.start[0] && p[0]<= n.end[0] &&
                                        p[1]>= n.start[1] && p[1]<= n.end[1] &&
                                        p[2]>= n.start[2] && p[2]<= n.end[2] &&
                                        p[3]>= n.start[3] && p[3]<= n.end[3]
                                        )
                                        break;
                        }
                        if( i== pStats->listIncludeFilter.end())
                        {
                                DWORD dwElapsed= GetTickCount() - dwStartTick;
                                pStats->dwSiteCountTick+= dwElapsed;

                                return 0;
                        }
                }
        }

        //=================================

        // Check Exclude Filters
        if( 1)
        {
                if( pStats->listExcludeFilter.size())
                {
                        unsigned long l;
                        unsigned char* p= (unsigned char*) &l;

                        l= inet_addr( pLogEntry->pHost);

                        std::list< CLogStats::IPFILTER>::iterator i;
                        for( i= pStats->listExcludeFilter.begin(); i!= pStats-
>listExcludeFilter.end(); i++)
                        {
                                CLogStats::IPFILTER& n= (*i);

                                if( p[0]>= n.start[0] && p[0]<= n.end[0] &&
                                        p[1]>= n.start[1] && p[1]<= n.end[1] &&
                                        p[2]>= n.start[2] && p[2]<= n.end[2] &&
                                        p[3]>= n.start[3] && p[3]<= n.end[3]
                                        )
                                        break;
                        }
                        if( i!= pStats->listExcludeFilter.end())
                        {
                                DWORD dwElapsed= GetTickCount() - dwStartTick;
                                pStats->dwSiteCountTick+= dwElapsed;
```

```
                    return 0;
                }
            }
        }

        //=================================

        DWORD dwTick= GetTickCount();

        //=================================

        pStats->nHits++;

        ...
        ...
        ...
}
```

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division



SCANNED,# 8

Application deficiencies found during scanning:

☐ Page(s)_____of_____ were not present for scanning. (Document title)

☐ Page(s)_____of_____were not present for scanning. (Document title)

☒ ***Scanned copy is best available.*** drawings